**IN THE SPECIFICATION:**

**(i) Please amend the paragraph on Page 9, line 23, through Page 10, line 23, as follows:**

The present invention is directed to data storage controllers that provide increased data storage/retrieval rates that are not otherwise achievable using conventional disk controller systems and protocols to store/retrieve data to/from mass storage devices. The concept of "accelerated" data storage and retrieval was introduced in ~~copending~~ U.S. Patent Application Serial No. 09/266,394, filed March 11, 1999, entitled "System and Methods For Accelerated Data Storage and Retrieval", which is now U.S. Patent No. 6,601,104 and ~~copending~~ U.S. Patent Application Serial No. 09/481,243, filed January 11, 2000, entitled "System and Methods For Accelerated Data Storage and Retrieval," which is now U.S. Patent No. 6,604,158, both of which are commonly assigned and incorporated herein by reference. In general, as described in the above-incorporated applications, "accelerated" data storage comprises receiving a digital data stream at a data transmission rate which is greater ~~that~~ **than** the data storage rate of a target storage device, compressing the input stream at a compression rate that increases the effective data storage rate of the target storage device and storing the compressed data in the target storage device. For instance, assume that a mass storage device (such as a hard disk) has a data storage rate of 20 megabytes per second. If a storage controller for the mass storage device is capable of compressing an input data stream with an average compression rate of 3:1, then data can be stored in the mass storage device at a rate of 60 megabytes per second, thereby effectively increasing the storage bandwidth ("storewidth") of the mass storage device by a factor of three. Similarly, accelerated data retrieval comprises retrieving a compressed digital data stream from a target storage device at the rate equal to, e.g., the data access rate of the target storage device and then decompressing the compressed data at a rate that increases the effective data access rate of the target storage device. Advantageously, accelerated data storage/retrieval mitigates the traditional bottleneck associated with, e.g., local and network disk accesses.

(ii) **Please amend the paragraph on Page 11, lines 1-13, as follows:**

Referring now to Fig. 1, a high-level block diagram illustrates a data storage controller 10 according to one embodiment of the present invention. The data storage controller 10 comprises a data compression engine 12 for compressing/decompressing data (preferably in real-time or psuedo real-time) stored/retrieved from a hard disk 11(or any other type of mass storage device) to provide accelerated data storage/retrieval. The DCE 12 preferably employs the data compression/decompression techniques disclosed in U.S. Serial No. 09/210,491 entitled "Content Independent Data Compression Method and System," filed on December 11, 1998, which is now U.S. Patent No. 6,195,024 which is commonly assigned and which is fully incorporated herein by reference. It is to be appreciated that the compression and decompression systems and methods disclosed in U.S. Serial No. 09/210,491 are suitable for compressing and decompressing data at rates, which provide accelerated data storage and retrieval. A detailed discussion of a preferred "content independent" data compression process will be provided below.

(iii) **Please amend the paragraph on Page 14, line 23, through Page 15, line 11, as follows:**

As discussed in greater detail below, upon host computer power-up or external user reset, the data storage controller 10 initializes the onboard interfaces 14, **15** prior to release of the external host bus 16 from reset. The processor of the host computer then requests initial data from the disk 11 to facilitate the computer's boot-up sequence. The host computer requests disk data over the Bus 16 via a command packet issued from the host computer. Command packets are preferably eight words long (in a preferred embodiment, each word comprises 32 bits). Commands are written from the host computer to the data storage controller 10 with the host computer as the Bus Master and the data storage controller 10 as the slave. The data storage controller 10 includes at least one Base Address Register (BAR) for decoding the address of a command queue of the data storage controller 10. The command queue resides within the cache 13 or within onboard memory of the DCE 12.

(iv) **Please amend the paragraph on Page 18, lines 10-22, as follows:**

The storage controller 20 further comprises computer reset and power up circuitry 28 (or "boot configuration circuit") for controlling initialization (either cold or warm boots) of the host computer system and storage controller 20. A preferred boot configuration circuit and preferred computer initialization systems and protocols are described in U.S. Patent Application Serial No. 09/775,897 ~~(Attorney Docket No. 8011-10)~~, filed concurrently herewith ~~(Express Mail Label No. EL679454245US)~~, which is commonly assigned and incorporated herein by reference. Preferably, the boot configuration circuit 28 is employed for controlling the initializing and programming the programmable logic device 22 during configuration of the host computer system (i.e., while the CPU of the host is held in reset). The boot configuration circuit 28 ensures that the programmable logic device 22 (and possibly other volatile or partially volatile logic devices) is initialized and programmed before the bus 16 (such as a PCI bus) is fully reset.

(v) **Please amend the paragraph on Page 22, lines 9-14, as follows:**

Fig. 3 illustrates another embodiment of a data storage controller ~~30~~ **35** wherein the data storage controller 35 is embedded within the motherboard of the host computer system. This architecture provides the same functionality as the system of Fig. 2, and also adds the cost advantage of being embedded on the host motherboard. The system comprises additional RAM and ROM memory devices 23a, 24a, operatively connected to the DSP 21 via a local bus 25a.

(vi) **Please amend the paragraph on Page 25, line 12, through Page 26, line 7, as follows:**

Referring now to Fig. 6, a flow diagram illustrates a method for initializing the programmable logic device 22 according to one aspect of the invention. In the following discussion, it is assumed that the programmable logic device 22 is always reloaded, regardless of the type of boot process. Initially, in Fig. 6a, the DSP 21 is reset by asserting a DSP reset signal (step 50). Preferably, the DSP reset signal is generated by the boot circuit configuration circuit 28 (as described in the above-incorporated U.S. Serial No. 09/775,897 ~~(Attorney Docket No. 8011-10)~~. While the DSP reset signal is asserted (e.g., active low), the DSP is held in reset and is initialized to a prescribed state. Upon deassertion of the DSP Reset signal, the logic code for the DSP (referred to as the "boot loader") is copied from the non-volatile logic device 24 into memory residing in the DSP 21 (step 51). This allows the DSP to execute the initialization of the programmable logic device 22. In a preferred embodiment, the lower 1K bytes of EPROM memory is copied to the first 1k bytes of DSP's low memory (0x0000 0000 through 0x0000 03FF). As noted above, the memory mapping of the DSP 21 maps the CE1 memory space located at 0x9000 0000 through 0x9001 FFFF with the OTP EPROM. In a preferred embodiment using the Texas Instrument DSP TMS320c6211GFN-150, this ROM boot process is executed by the EDMA controller of the DSP. It is to be understood, however, that the EDMA controller may be instantiated in the programmable logic device (Xilinx), or shared between the DSP and programmable logic device.

(vii) **Please amend the paragraph on Page 37, lines 14-19, as follows:**

The DSP services request for its external bus from two requestors, the Enhanced Direct Memory Access (EDMA) Controller and an external shared memory device controller. The DSP can typically utilize the full 280 megabytes of bus bandwidth on an 8k through 64K byte (2k word through 16k word) burst basis. It should be noted that ~~the DSRA does not utilize~~ the SDRAM memory is not utilized for interim processing storage, and as such ~~only utilizes~~ bandwidth is only utilized in direct proportion to disk read and write commands.

(viii) **Please amend the paragraph on Page 41, lines 12-18, as follows:**

Once the data is preloaded, when the computer system bus issues its first read commands to the data storage controller seeking operating system data, the data will already be available in the cache memory of the data storage controller. The data storage controller will then be able to instantly start transmitting the data to the system bus. Before transmission to the bus, if the **data** was stored in compressed format on the boot device, the data will be decompressed. The process of preloading required (compressed) portions of the operating system significantly reduces the computer boot process time.

(ix) **Please amend the paragraph on Page 49, line 21, through Page 50, line 12, as follows:**

Again, it is to be understood that the embodiment of the data compression engine of Fig. 9 is exemplary of a preferred compression system which may be implemented in the present invention, and that other compression systems and methods known to those skilled in the art may be employed for providing accelerated data storage in accordance with the teachings herein. Indeed, in another embodiment of the compression system disclosed in the above-incorporated U.S. Patent No. 6,195,024 ~~Serial No. 09/210,491~~, a timer is included to measure the time elapsed during the encoding process against an *a priori*-specified time limit. When the time limit expires, only the data output from those encoders (in the encoder module 125) that have completed the present encoding cycle are compared to determine the encoded data with the highest compression ratio. The time limit ensures that the real-time or pseudo real-time nature of the data encoding is preserved. In addition, the results from each encoder in the encoder module 125 may be buffered to allow additional encoders to be sequentially applied to the output of the previous encoder, yielding a more optimal lossless data compression ratio. Such techniques are discussed in greater detail in the above-incorporated U.S. Patent No. 6,195,024 ~~Serial No. 09/210,491~~.

(x) **Please amend the paragraph on Page 50, lines 13-20, as follows:**

Referring now to FIG. 10, a detailed block diagram illustrates an exemplary decompression system that may be employed herein or accelerated data retrieval as disclosed in the above-incorporated U.S. Patent No. 6,195,024 ~~Serial No. 09/210,491~~. In this embodiment, the data compression engine 180 retrieves or otherwise accepts compressed data blocks from one or more data storage devices and inputs the data via a data storage interface. It is to be understood that the system processes the input data stream in data blocks that may range in size from individual bits through complete files or collections of multiple files. Additionally, the input data block size may be fixed or variable.

(xi) **Please amend the paragraph on Page 51, line 20, through Page 52, line 6, as follows:**

As with the data compression systems discussed in U.S. Patent No. 6,195,024 ~~Application Serial No. 09/210,491~~, the decoder module 165 may include multiple decoders of the same type applied in parallel so as to reduce the data decoding time. An output data buffer or cache 170 may be included for buffering the decoded data block output from the decoder module 165. The output buffer 70 then provides data to the output data stream. It is to be appreciated by those skilled in the art that the data compression system 180 may also include an input data counter and output data counter operatively coupled to the input and output, respectively, of the decoder module 165. In this manner, the compressed and corresponding decompressed data block may be counted to ensure that sufficient decompression is obtained for the input data block.